SIAM WORKSHOP ON INTEGRATION OF CAD AND CFD, UNIVERSITY OF CALIFORNIA DAVIS, DAVIS CA

On the Use of CAD-Native Predicates and Geometry in Surface Meshing

M. J. Aftosmis
NASA Ames Research Center
Moffett Field, CA 94035

April 12-13, 1999

National Aeronautics and
Space Administration
Ames Research Center

On the Use of CAD-Native Predicates and Geometry in Surface Meshing

M. J. Aftosmis[†]
NASA Ames Research Center
Moffett Field, CA 94035

Abstract

Several paradigms for accessing CAD geometry during surface meshing for CFD are discussed. File translation, inconsistent geometry engines and non-native point construction are all identified as sources of non-robustness. The paper argues in favor of accessing CAD parts and assemblies in their native format, without translation, and for the use of CAD-native predicates and constructors in surface mesh generation. The discussion also emphasizes the importance of examining the computational requirements for exact evaluation of triangulation predicates during surface meshing.

The *native* approach is demonstrated through an algorithm for the generation of closed manifold surface triangulations from CAD geometry. CAD parts and assemblies are used in their native format, and a part's native geometry engine is accessed through a modeler-independent application programming interface (API). In seeking a robust and fully automated procedure, the algorithm is based on a new physical space manifold triangulation technique specially developed to avoid robustness issues associated with poorly conditioned mappings. In addition, this approach avoids the usual ambiguities associated with floating-point predicate evaluation on constructed coordinate geometry in a mapped space. The technique is incremental, so that each new site improves the triangulation by some well defined quality measure. The algorithm terminates after achieving a prespecified measure of mesh quality and produces a triangulation such that no angle is less than a given angle bound, α, or greater than $\pi - 2\alpha$. This result also sets bounds on the maximum vertex degree, triangle aspect-ratio and maximum stretching rate for the triangulation. In addition to the output triangulations for a variety of CAD parts, the discussion presents related theoretical results which assert the existence of such an angle bound, and demonstrate that maximum bounds of between 25° and 30° may be achieved in practice.

1. Introduction

MESH generation has long been recognized as a bottleneck in the CFD process.[1] The last decade has witnessed a myriad of international and domestic conferences and symposiums aimed at focusing research on this impediment. Unstructured, hybrid, and Cartesian mesh methods are all aimed at simplifying the mesh generation task for complex configurations. The success of these approaches is well represented in the literature and with an appropriate initial surface triangulation, the volume mesh generation can generally be accomplished in a relatively automated fashion in minutes-to-hours on an engineering workstation.[2-7] As faster processors continue to shrink the wall-clock time required for both mesh generation and flow solution, the man-hour intensive task of extracting an initial surface discretization from a CAD geometry promises to become an ever larger fraction of the bottleneck. Additionally, if the user must be involved in the extraction of surface data from CAD, then mesh adaptation - which involves enriching the discretization on the body surface - will remain an elusive goal.

Historically, surface discretization has been one of the least automated steps in the numerical simulation cycle, and for good reason. Due to its dependence on implicitly defined surfaces and curves, CAD data is by its nature imprecise. Various geometry engines typically demonstrate discrepancies in their interpretations of the same entities. As a result, "repair" of CAD surfaces has become an area of substantial research.[8][9] This problem is exacerbated when CAD models are output in many of the standard formats, since such files frequently do not include important topological and construction information along with the entity geometry.

In response to these and other requirements for user assistance, many in the research and industrial CFD communities have adopted an interactive paradigm for surface mesh generation. The commercial unstructured mesh generators in Refs. [7],[10] and [11] all interact with CAD data through files which have been translated from their CAD native environment to some standardized format (namely IGES[12], STEP[13] or STL[14]).[1]

This paper examines an alternative paradigm. The approach interfaces with the CAD system in a dynamic manner via calls to CAD native routines. By accessing the model in its native environment, this approach avoids translation to a format which can deplete the model of topological information. This is important since it avoids the consistency conflicts that can occur when two different geometry engines attempt to infer topological information from imprecise data.

To avoid placing CAD specific calls in the software, we argue in favor of wrapping these calls in a standardized Application Programming Interface (API) such as CAPRI.[15] This library presents a standardized interface to the application program for various CAD systems.[2] CAPRI supports a variety of operations like truth testing, geometry construction, and entity queries. This strategy also divides the task of software maintenance between tasks associated with the CAD system and those associated with the surface mesher.

Maintaining the consistency of the models by direct manipulation of CAD parts and assemblies is the first step that this work takes toward building a robust method for surface triangulation. A basic premise of this approach is that we resolve consistency problems on as simple a model as possible, and

[†] Research Scientist, NASA Ames, *aftosmis@nas.nasa.gov.*

[1] Recent releases of some of this software now supports "direct" interfaces which do read parts in their native formats, however, this practice is not the norm.

[2] CAPRI currently supports ProEngineer™ Unigraphics™ and SDRC I-DEAS™ with CATIA™ support in beta test.

maintain this consistency as the triangulation evolves and becomes more complex.

1.1. Abstract Geometric Structures

Following the approach of Yap,[16] our approach toward generating a robust geometric algorithm contends that a geometric structure, D, consists of four elements:[2]

$$D = \left(G, \lambda, \Phi(\mathbf{z}), \mathbf{z}\right) \qquad (1)$$

Where the graph, $G = (V, E)$ is a directed set of vertices, V, and edges, E. λ is a function describing the index labels of the graph. Φ is a geometric operator which represents the consistency predicates for the connectivity and is a function of the actual coordinates $\mathbf{z}$. G represents a tessellation of the vertices and is therefore purely combinatorial. A structure is said to be *consistent* if the predicate $\Phi(\mathbf{z})$ holds.

As an example, a 2-D Delaunay triangulation algorithm usually makes use of an *inCircle* predicate[17], $\Phi_{inCircle}$ which establishes G by insisting that the circumcircle of the triangle $\Delta_{a,b,c}$ can contain no other vertex in the graph. If such a predicate holds for every triangle in G, then this *instance* of the geometric structure D is said to be consistent. This interpretation offers direct insights into the formulation of a robust algorithm for creating triangulations of CAD volumes.

1.2. Robustness

Consistent CAD Models

The rational B-splines used to describe surfaces in most CAD systems are implicitly defined for physical space coordinates of the geometry. Therefore, the *constructors* for vertex geometry generally require a Newton solve carried to some internal tolerance. Since the results of this construction will be subject to both tolerance and round-off error, the system may then "nudge" the constructed point, z_i, to some nearby exactly representable location (on an integer grid, for example). If the geometry engine's predicate, $\Phi_{onSurf}(z, S)$, for determining if z_i is on a surface, S is consistent then it will return "true" when later queried if z_i lies on the surface. However, if $\tilde{\Phi}_{onSurf}(z, S)$ now represents some user-defined predicate which may be ignorant of the systems construction rules, then it is very unlikely to return consistent results.

The CAPRI API ensures the maintenance of a consistent representation of the model by providing access a subset of the CAD geometry engine's constructors, queries and predicates. Our implementation adopts a multi-threaded programming approach which runs the geometry engine on its own thread in order to respond to queries from the main triangulation thread.

Physical Space Triangulation

A variety of existing surface meshing techniques adopt a *mapped-space* approach for generating surface triangulations. In this approach, a surface and its bounding curves are triangulated in a 2-D parameter space, which may or may not have some additional scaling imposed. Ref.[18] provides a mathematical description of the Non-Uniform Rational B-Spline (NURBS) surfaces typically used in CAD systems. Here we note only that an iterative method is required to solve for the physical space coordinates of a position specified on the surface in the parameter space. This process involves division of two (generally) high-order polynomials, and is therefore subject to both error associated with finite-precision arithmetic and error associated with tolerancing for the convergence of the iterative solve. As a result, computed coordinates in the mapped space are necessarily noisy and cannot be considered exact values. Two consequences of this approach are:

1. Since the error bounds on the input, $\mathbf{z}$, are unknown, evaluation of the triangulation predicates (e.g. $\tilde{\Phi}_{inCircle}$) are unlikely to robustly produce consistent results (see Fig. 19 Ref.[17], also [19] and [20]).
2. The polynomial basis for the NURBS may be high-order, and therefore small errors in parameter space may produce dramatic results in physical space - even within the subspace for which the surface is defined. The likelihood of encountering poorly conditioned mappings is the primary reason that CAD repair software generally attempts to re-normalize the NURBS surface and recast it using basis polynomials with as low an order as possible[8].

These two observations motivate an examination of physical space triangulation techniques. In this approach, we construct a manifold triangulation on the surface, and evaluate the triangulation predicates in $\Re^3$. New sites are constructed by the CAD geometry engine and, since this output is consistent with the system's internal predicates, it is considered exact by the external predicates of the triangulation algorithm. The presentation in §3 emphasizes both minimization and tracing of the floating-point error in evaluation of the triangulation predicates. Computational requirements for exact evaluation are presented.

2. The CAPRI API

Our basic approach is to take a crude manifold triangulation of each closed volume in the CAD assembly, and improve it until it satisfies a preset measure of mesh quality, or produces a preset number of triangles. A variety of mesh quality measures may be defined within this framework, and this preliminary investigation examines two such criterion: (1) the mesh must be free from small angles (sliver triangles); (2) edges in the triangulation must not deviate from the underlying model by more than a prescribed tolerance.

2.1. CAPRI Volumes

CAD entities are accessed through the CAPRI programming interface.[15] This API provides a layer of indirection such that CAD system specific data may be accessed by an application program using CAD system neutral function calls.

[2] Ref.[16] actually writes eq.(1) as $D = (G, \lambda, \Phi(\mathbf{z}), I)$ where I is a mapping from the input parameters $\mathbf{c}$ to $\mathbf{z}$, $I\colon \mathbf{z} \to \mathbf{c} = (c_1, \ldots, c_n) \in \Re^d$.

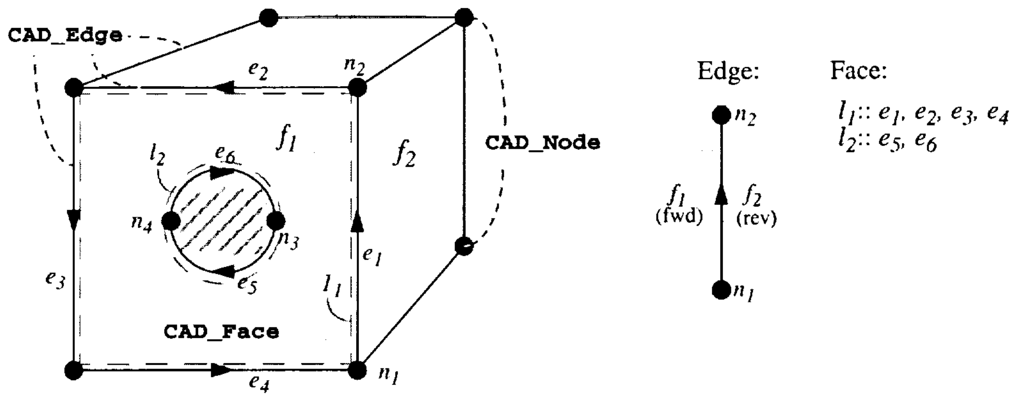


Figure 1. CAPRI data structured demonstrated on a simple volume with a cylindrical cutout.

Figure 1 presents an abstract view of the entities that CAPRI provides. A *cad_node* is the lowest dimensional entity and corresponds to a point in 3-space. A *cad_edge* has a cad_node at both ends. Each is directed from its origin, O, to its destination, D. Cad_edges are not assumed to be simplicial and may follow a general curve in space (see e_5 and e_6 in Fig.1). Each edge is connected to two *cad_face* entities. In general, these faces are composed of several loops and are not assumed planar, since they follow the underlying parameterization of the surface. A cad_face is composed of one or many *loops*, which are collections of oriented edges. Loops are oriented such that the surface of the cad_face lies inside them when they are traversed in a counterclockwise circuit when viewed from a point outside the solid. This convention permits holes in a surface to be described by a clockwise loop. In Figure 1, cad_face, f_1, consists of loops l_1 and l_2, each of which is composed of cad_edge entities. The edge ordering of l_2 indicates that it is clockwise, and therefore describes a hole in f_1. Edges and faces have an underlying parameterization, and while points may be queried for their parametric values (u, v), details of this ruling are not otherwise exposed to the application program.

2.2. Initial Manifold Triangulation

A central theme in the present approach is the maintenance of a closed volume throughout the procedure. For each volume, CAPRI returns a simplicial decomposition of each of the m cad_face entities, S_i, where $S_i \in \{S_1, S_2, \ldots, S_m\}$. Each of these triangulations are manifold within their respective cad_edges. In addition, an indexing function, λ_C, is returned for each cad_volume. Therefore a simplicial, manifold representation of each cad_volume, S_C may be constructed by taking the union of the decompositions of all the cad_face entities of a volume, subject to the indexing λ_C.

$$S_C = \bigcup_{i=1}^{m} S_i \tag{2}$$

Figure 2 displays an example of this initial triangulation for a simple part. The manufacturing die shown has 14 cad_face entities and the initial triangulation, S_C, has 270 triangles. As is typical, this triangulation is quite irregular, and planar regions are decomposed into as few triangles as possible. Extremely high aspect ratio triangles are common in these boundary triangulations. Figure 2.b labels selected CAD entities on this triangulation. Notice that although some cad_face sites may be present, this initial triangulation is essentially a boundary triangulation and the number of triangles is proportional to the number of cad_edges.

Despite the poor quality of the triangulation, the structure in Fig.2 has several desirable properties. Namely, it is consistent, manifold, oriented and closed. We wish to improve this triangulation by adding sites on both the cad_edge and cad_face entities and by enforcing an external predicate governing the type of triangulation.

3. Mesh Improvement

Our approach for manifold surface triangulation traces its roots to work on *quality triangulations* of Planar Straight

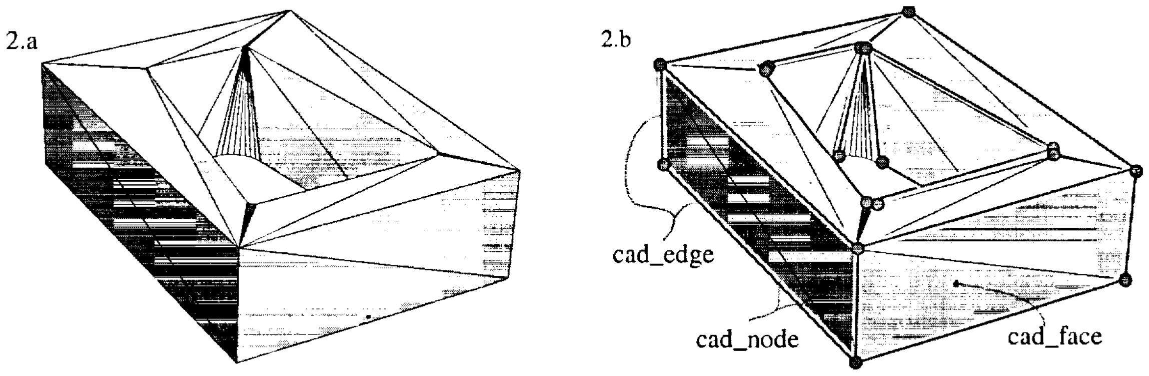


Figure 2. Initial closed, manifold, surface triangulation, S_C, of a CAD model for a manufacturing die. Underlying CAD entities exposed to application program are labeled in the frame on the right.

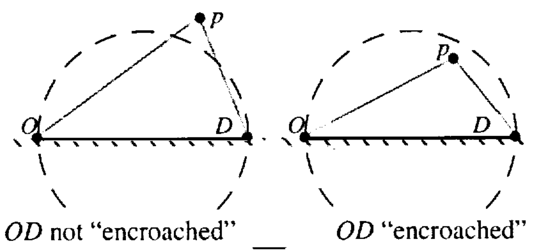


Figure 3. Constraining edge $\overline{OD}$ and its diametral circle. The edge is "encroached" if any site, p, falls within the *diametral circle* of $\overline{OD}$

Line Graphs (PSLGs)[21][22][23] and related work on quality triangulations of manifold surfaces.[24] Work in this field began with the efforts of Ref.[21] which presented an algorithm with both shape and size guarantees. The resulting meshes were size-optimal and had no triangle with an aspect ratio greater than 5. In this context, the *aspect ratio AR,* of a triangle, is defined as the length of the longest edge divided by that of the shortest one. One can show that if α is the smallest angle of a triangle, then

$$\frac{1}{|\sin\alpha|} \le AR \le \frac{2}{|\sin\alpha|}. \tag{3}$$

Therefore α is frequently used to describe the quality of a given triangle.

Before presenting the manifold triangulation technique, this section first recounts a related algorithm for quality triangulation for PSLGs from Ref.[23]. It then presents a fundamentally similar algorithm for triangulating curved surfaces and notes which aspects of the PLSG method have been relaxed in the extension.

3.1. Quality Triangulation of PSLGs

While the manifold surface triangulation technique of Ruppert[23] and the PSLG method of Chew[24] are similar in many respects, our manifold technique follows Ruppert's approach more closely. Section 3.2 addresses some of the reasoning behind this choice.

An essential feature of the algorithm is the notion of an encroached constraining edge. As illustrated in Figure 3, a constraining edge, $\overline{OD}$, is said to be *encroached upon* if any other site (visible to $\overline{OD}$) lies within the diametral circle of the edge.

If one recalls that the circumcenter of a right triangle falls on the hypotenuse, then its easy to show that for a triangulation which is Delaunay or locally maxmin, a predicate for encroachment may be formulated as a vector dot product. Thus for similarly sized[1] floating-point data with p-bit significands, this predicate can be evaluated exactly in a register with a $2p$-bit significand[20]. In a practical sense, this implies that as long as the edges are small by comparison to their distance from the origin, this predicate will be exact if computed in double-precision, using single-precision data.

1. The qualifier "similarly sized" is necessary to guard against the case where a coordinates of one point is less than half that of another. Extended precision would be required in such a case.

In the algorithms, $\tilde{\Phi}_{Encroached}(e)$ denotes application of this predicate to an edge, e. The ($\tilde{\ }$) superscript reminds us that since this predicate is part of our triangulation algorithm, it is not native to the CAD system.

The presentation of Ref.[23] recovers the constraining edges of the triangulation as the algorithm advances. To clarify its relation with our manifold triangulation technique, we recast the original algorithm assuming that it begins with a constrained boundary triangulation of the input vertices, V, of the constraint edges. Furthermore, this initial triangulation is assumed to be the constrained Delaunay triangulation of the input sites, $\mathcal{CDT}(V)$.

The algorithm is quite elegant in that it consists of only two major operations:

1. Split a constrained edge: Add a site near[2] the mid-point of a constraining edge, and replace the original edge with the two new edges in the constraint list.
2. Split a triangle: Add a site to the circumcenter of a triangle, t. Note that if the triangle is obtuse, this site will not fall within t.

Algorithm Q: *Quality triangulation of a PLSG*

```
  Input:      Planar Straight Line Graph, $X$, with input vertices, $V_{in}$.
              Target angle, $\alpha$.
 Output:      $\mathcal{CDT}(V_{out})$ with all angles $\ge \alpha$.
Initialize:   Compute $\mathcal{CDT}(V_{in})$. Build minimum angle priority
                 queue, $PQ_{min}$ with $t_{PQ}$ denoting triangle at head of
                 queue, having min angle $\theta_{PQ}$
1. Apply $\tilde{\Phi}_{Encroached}(e)$ to all constraint edges:
      While(any constraining edge is encroached){
         Split constrained edge. Update $\mathcal{CDT}(V)$, Update $PQ_{min}$.
      }
2.    While ($\theta_{PQ} < \alpha$){
2.a      Let $p$ be the circumcenter of $t_{PQ}$.
2.b      If ($p$ encroaches any constraining edge, $e$)
2.c         Split constrained edge. Update $\mathcal{CDT}(V)$, Update $PQ_{min}$.
2.d      Else Split triangle $t_{PQ}$:
            Add $p$ to $V$. Update $\mathcal{CDT}(V)$, Update $PQ_{min}$.
      }
3.    Output $\mathcal{CDT}(V)$.
```

While simple, Ref.[23] proves that Alg.Q produces triangulations with the following desirable properties.

1. Quality: An angle bound 20.7° is guaranteed, and values as high as 30° may be achieved in practice.

2. Output Size: The size of the output triangulation is within a constant of the optimal number of triangles required to satisfy the angle criteria.

3. Size Optimality: Small input constraint edges are surrounded by proportionally small triangles. Nearby triangles have similar sizes, and the size variation of triangles in the mesh is proportional to the distance between them.

3.2. Difficulties on Curved Surfaces

In Ref.[24], Chew presents a quality triangulation technique which is closely related to Alg.Q. This work raises a number

2. We use the modified insertion strategy presented in Ref.[23].

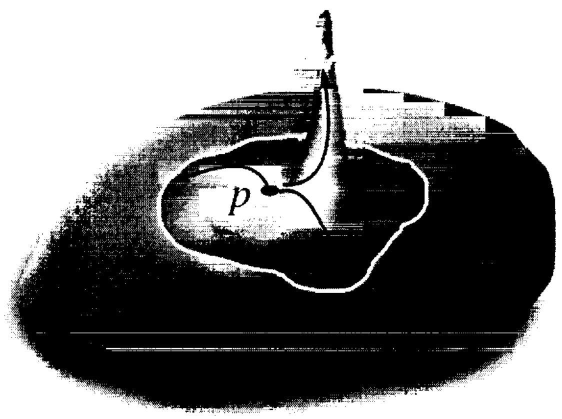


Figure 4. An example of non-intuitive consequences of a straightforward interpretation of the inCircle PSLG predicate on curved surfaces. With distances measured using the geodesic distance along the surface, the loci of points equidistant from p reaches around the spire but does not include its tip.

of difficulties associated with the extension of the PSLG method to curved surfaces.

Alg.Q has two salient aspects. (1) The triangulation is constrained Delaunay. (2) New sites are added at circumcenters. Chew observes that a straightforward definition of a circle on a surface is the loci of points on the surface which are equidistant from another point on the surface, where all distances are measured using the geodesic distance along the surface. While straightforward, this definition is problematic. Distances along the surface must be measured in physical space, and will therefore be expensive to compute on NURBS surfaces. In addition, due to the inherent error in finding the coordinates of a point on such a surface, robust predicates based upon this definition will be difficult to formulate. Finally, Chew notes that this definition has less subtle and non-intuitive consequences. As shown in Figure 4, a circle whose center lies near the base of a sharp spire, for example, may reach completely around the spire without also including the tip of the spire.

To circumvent such difficulties, the method in [24] makes use of an alternative definition of a circle. In the plane, the three vertices of a triangle define a unique circle. In 3-dimensions, however, an infinite family of spheres may be passed through those three points. Connecting the line through the centers of this family of spheres and intersecting this line with the surface identifies a particular sphere in this family. The circumcenter of the triangle may be defined to be the loci of points at the intersection of this particular sphere and the surface. Once this sphere is found, then the $\Phi_{inCircle}$ triangulation predicate may be evaluated by simply computing distances in three dimensions as a vector magnitude.

Despite the effort, problems still exist with this predicate. (1) If the triangulation does not resolve the underlying surface closely enough, the line of circumsphere centers will not necessarily intersect the surface. Alternatively, it may also intersect the surface in multiple locations places. (2) Computing the intersection of this line with the surface will require an iterative method and will therefore be computationally intensive. (3) Once this intersection is successfully located on the surface, one must determine which triangle the point falls inside. Since the triangulation only matches the surface at the vertices, ambiguous situations may arise when two triangles claim ownership of the same site (near a ridge, for example). (4) On a curved surface, the circumcenters of two triangles with a shared edge may not be consistent. Specifically, when $\Phi_{inCircle}$ tests one of the triangles against the opposite vertex of the other, the results may not agree when the roles of the triangles are reversed. Lemma 5 in Ref.[24] shows this situation will arise if the normal vectors of the triangles vary by more than $\pi/2$. A successful algorithm must guard against $\Phi_{inCircle}$ becoming inconsistent.

3.3. A Physical-Space Surface Meshing Algorithm

These outstanding ambiguities and computational expense motivated a search for a more manageable algorithm. Our method makes two fundamental changes.

- To avoid the ambiguity associated with the definition of $\Phi_{inCircle}$, we do not attempt a Delaunay triangulation of the curved surface. Instead we seek a triangulation which is everywhere locally maxmin. While this may seem to constitute a dramatic relaxation, recall that the goal is a practical algorithm, and we have no reason to prefer strictly Delaunay output triangulations. In addition, since one property of a Delaunay triangulation is that it is maxmin, this choice is worth investigating.
- When all angles of a triangle are acute, the circumcenter falls within the triangle. Ownership of the new site can then be uniquely assigned to this triangle. However, when a triangle is obtuse, ownership can become less clear. Therefore we make a simple choice: When a triangle is obtuse, we insert the new site at the centroid of the *other* triangle sharing the edge opposite the obtuse angle. Figure 5. illustrates this insertion rule. If t_i is an obtuse triangle, then the new sites are added at the centroid of t_{opp}.

While admittedly *ad hoc*, the modified insertion strategy is not as arbitrary as it may initially seem. As a particular angle of t_i opens up in the transition from acute to obtuse, the circumcenter will pass from within t_i to within t_{opp}. The centroid of t_{opp} may therefore be thought of as an *approximate*

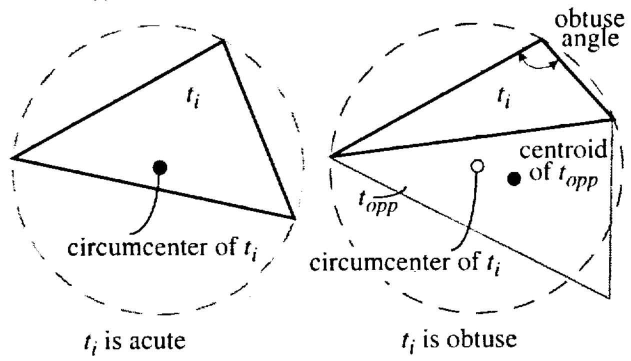


Figure 5. Sites are added at the centroid of a triangle if the triangle is acute. Otherwise they are added at the centroid of the triangle opposite the obtuse angle.

circumcenter of t_i. Denoting the radius of the circumcircle of t_i as R, it can be shown that if one chooses an approximate circumcenter within a distance fR of the true circumcenter, then an angle bound of

$$\alpha \le \operatorname{asin}\left(\frac{1-f}{2}\right) \quad (4)$$

may still be achieved. Ref. [24] contends that even using approximate circumcenters, angle bounds of 30° may be achieved in practice.

With these changes, the manifold triangulation algorithm becomes:

Algorithm M:

Quality Manifold triangulation of a CAD volume.

```
     Input: Underlying CAD volume, $P$, with initial triangulation
             $S_C$, and input vertices, $V_{in}$.
            Target angle, $\alpha$.
    Output: $\mathcal{CXN}(V_{out})$ with all angles $\ge \alpha$.
Initialize: Compute constrained locally maxmin triangulation
             $\mathcal{CXN}(V_{in})$, using cad_edge entities as constraints.
             Build minimum angle priority queue, $PQ_{min}$ with $t_{PQ}$
             denoting triangle at head of queue, having min angle
             $\theta_{PQ}$
1. Apply $\tilde{\Phi}_{Encroached}$ to all constraint edges:
     While(any constraining edge is encroached){
         Split constrained edge. Update $\mathcal{CXN}(V)$, Update $PQ_{min}$.
     }
2.   While ($\theta_{PQ} < \alpha$){
2.a      If ($t_{PQ}$ is not obtuse)
              Assign $t := t_{PQ}$, with circumcenter $p$.
         Else Assign t:= $t_{opp}$ with centroid $p$.
2.b      If ($p$ encroaches any constraining edge, $e$)
2.c          Split constrained edge. Update $\mathcal{CXN}(V)$, Update $PQ_{min}$.
2.d      Else Split triangle $t$:
             Add $p$ to $V$. Update $\mathcal{CXN}(V)$, Update $PQ_{min}$.
     }
3.   Output $\mathcal{CXN}(V)$.
```

When the algorithm terminates, all angles are greater than α. Thus, it recovers the properties of quality and size-optimality cited after the presentation of Alg.Q in §3.1. The bound on output size, however, depends on the site insertion strategy always inserting new vertices within the circumcircle of t_{PQ},[23] and the modified strategy will not always guarantee this, thus the algorithm sacrifices strict proof of this property.

The new algorithm requires initialization with a transformation of the part's original constrained manifold triangulation S_C to one which is everywhere locally maxmin. If we assume the existence of a triangulation predicate $\tilde{\Phi}_{XN}$ which can be enforced for every pair of triangles sharing an edge in the triangulation (similar to the application of $\tilde{\Phi}_{inCircle}$), then this initialization may be performed with edge sweeps followed by edge-swapping when a violation is encountered. While the possibility of multiple sweeps makes this a seemingly inefficient approach, we recall that the initial complexity of S_C is only proportional to the number of cad_edges in the geometric structure. Thus this simplistic approach is not a problem.

Site insertion proceeds in a manner comparable to the PSLG algorithm. A convenient implementation makes use of the incremental insertion strategy of Ref.[25]. However, since the triangulation is no longer Delaunay, both forward and reverse propagation is necessary after edge swaps. Sites from edge or triangle splitting must be projected to their actual locations on the underlying surface, and the native constructors are used for this through the CAD API. Ref.[25] gives a modified point placement strategy for splitting encroached edges. Our implementation of Alg.M adopts this strategy without modification.

3.4. Triangulation Predicates

Algorithm M rests on two new triangulation predicates. The first tests a triangle for an obtuse angle, $\tilde{\Phi}_{ob}(t)$ and the second, $\tilde{\Phi}_{XN}(e)$, tests if the edge, e, shared by any two triangles maximizes the minimum angle in both triangles. This is accomplished by comparison with a swapped edge, e', which connects the opposite vertices of the two triangles. Our approach hinges on the hope of evaluating these predicates robustly in physical space.

Both of these predicates can be formulated with direct measurement of the angles in a mesh. Since a triangle has three points, it uniquely defines a plane in three dimensions. Angle measurement in a plane is unambiguous - despite the fact that the triangles form a discrete manifold which is of lower dimension than the surrounding space. Numerically, we recall that since all vertex locations returned by the CAD engine are considered exact, the error bound on the input is identically zero.

$\tilde{\Phi}_{ob}(t)$ is applied by a logical *or* accumulation of $\tilde{\phi}_{\angle j-1,j,j+1}$, where j is a cyclic index running over the vertices of t, $j \in \{1,2,3\}$. This angle predicate is formed by comparison of the square of the edge opposite j to that of a hypothetical edge formed by assuming that the edges of t incident on j form a right angle in the plane of t.

$$\tilde{\Phi}_{ob}(t_{123}) = \tilde{\phi}_{ob}(\angle 123) \vee \tilde{\phi}_{ob}(\angle 231) \vee \tilde{\phi}_{ob}(\angle 312) \quad (5)$$

where

$$\tilde{\phi}_{\angle j-1,j,j+1} = \begin{cases} T \text{ if } |v_j - v_{j-1}|^2 + |v_{j+1} - v_j|^2 > |v_{j+1} - v_{j-1}|^2 \\ F \qquad \text{otherwise} \end{cases} \quad (6)$$

Notice that Eq.(6) requires only subtraction and multiplication of data which is known exactly. Thus the computational requirements for exact evaluation are the same as for evalua-

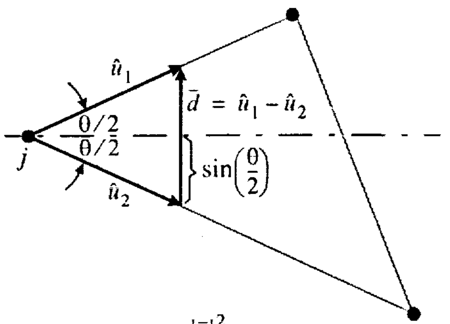

$\hat{u}_1$
$\bar{d} = \hat{u}_1 - \hat{u}_2$
$\theta/2$
$\theta/2$
j
$\sin\left(\frac{\theta}{2}\right)$
$\hat{u}_2$

Figure 6. Construction for $|\bar{d}|^2$, the square of the magnitude of the difference of unit vectors incident on vertex j.

tion of the dot product for $\tilde{\Phi}_{Encroached}(e)$ in §3.1.

Evaluation of $\tilde{\Phi}_{XN}(e)$ requires a more direct method of angle measurement. Figure 6 shows a construction for this measurement. Recalling that *sin*( ) is monotone over the interval $[0, \pi/2]$ the construction in the figure shows that the difference of the unit vectors of the edges incident upon any vertex, *j*, is sufficient to define a vector $\bar{d}$ whose magnitude varies monotonically with the angle formed by the edges incident upon *j*. As in the preceding two predicates, computationally, it is sufficient to evaluate only the square of this magnitude.

The computational requirements for this predicate are somewhat less simple. The computation of unit vectors requires robust computation of the inverse of a vector magnitude $1/|\bar{V}|$. Thus, exact evaluation of $\tilde{\Phi}_{XN}(e)$ requires software arithmetic, like the packages in Refs.[26], [27] or [28]. Although this need must be viewed as a drawback, we note that it is confined to a single predicate, and to a relatively simple expression. Moreover, since the input geometry is exact, exact computation remains a feasible strategy. In the preliminary results shown in §4, all computation was performed using only double-precision floating- point hardware, and the option of software arithmetic was not pursued.

3.5. Edge Refinement

Algorithm M drives small angles out of the evolving triangulation. In addition we wish to satisfy some edge-based criterion like a chord-height tolerance. After initially creating a triangulation free of small angles we apply an edge-refinement procedure to enforce such requirements.Algorithm E considers a generic edge-based scalar $\gamma(e)$. In our implementation *chord-height* is defined as the square of the distance from the middle of an edge to the corresponding location on the actual surface of the model (provided through the CAD API by the geometry engine).

Algorithm E: Edge Refinement of Manifold Triangulation.

```
   Input:    Underlying CAD volume, $P$, with current triangulation,
             $\mathcal{CXM}(V)$, and vertex set, $V$.
             Edge criteria $\gamma$.
  Output:    $\mathcal{CXM}(V_{out})$ with all edges satisfying $\gamma(e) < \gamma$.
Initialize:  Build priority queue, $PQ_\gamma$ with $e_{PQ}$ denoting edge at
             head of queue, having $\gamma(e_{PQ})$.
1. Apply $\tilde{\Phi}Encroached$ to all constraint edges:
     While(any constraining edge is encroached){
        Split constrained edge. Update $\mathcal{CXM}(V)$, Update $PQ_\gamma$.
     }
2.   While ($\gamma(e_{PQ}) > \gamma$){
2.a     Let $p$ be midpoint of $e$.
2.b     If ($p$ encroaches any constraining edge, $e$)
2.c        Split constrained edge. Update $\mathcal{CXM}(V)$, Update $PQ_\gamma$.
2.d     Else Split edge $e$:
           Add $p$ to $e$. Update $\mathcal{CXM}(V)$, Update $PQ_\gamma$.
     }
3. Output $\mathcal{CXM}(V)$.
```

Assuming that $\gamma(e)$ is a static criterion, like a chord-height tolerance, Alg.M can then be re-applied to $\mathcal{CXM}(V)$ to remove any small angles created during edge refinement and swapping.

4. Results and Discussion

This section presents example meshes on several CAD parts from a variety of sources. All the example parts were read in their native CAD file format using the CAPRI API without special treatment. The investigations focus on examination of issues raised in the presentation of the triangulation algorithm in §3.3 and the edge refinement strategy from §3.5.

4.1. Minimum Angle Bound

In §3.3, Algorithm M was presented without firm proof of termination. Moreover, the discussion noted that the modified site insertion strategy for obtuse triangles violates one of the assumptions that establishes a bound on the output size of the mesh in the PSLG method. It is therefore necessary to demonstrate the performance of Alg.M to show that it both terminates and produces meshes with an economy similar to that of the PSLG method upon which it is based.

Figure 7 contains a histogram of the evolution of the smallest angle in the mesh as Alg.M proceeds on the manufacturing die example problem used in earlier illustrations. While this curve is far from monotone, it clearly displays the steady improvement of the minimum angle in the mesh as the algorithm proceeds. The steep initial rise indicates rapid annihilation of extremely small angles in the mesh, and the mesh achieves a minimum angle of almost 29° by the end of the histogram.

The dashed line at 25° highlights the first time that all angles in the mesh exceeded this value. Tracing this value on the abscissa shows that setting the angle bound, α, to 25° will cause Alg.M to terminate after generating 2606 triangles. Figure 8 shows the resulting triangulation. As discussed in §3.1 and §3.3, the presence of an angle bound ensures that small features are surrounded by proportionally small trian-

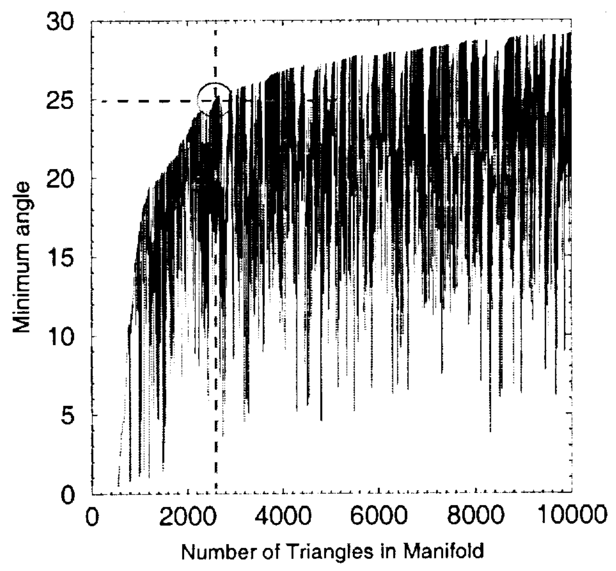


Figure 7. Histogram of minimum angle during mesh evolution using Alg.M (without the edge refinement of §3.5) for the manufacturing die example presented earlier. In this example, a mesh with a minimum angle of 25° would contain 2606 triangles.

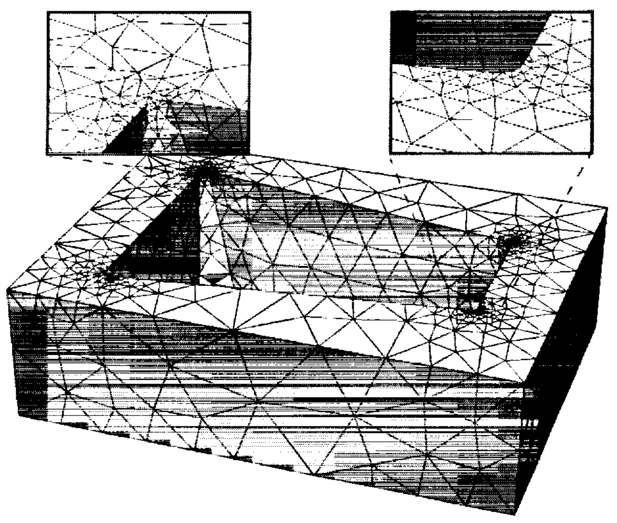
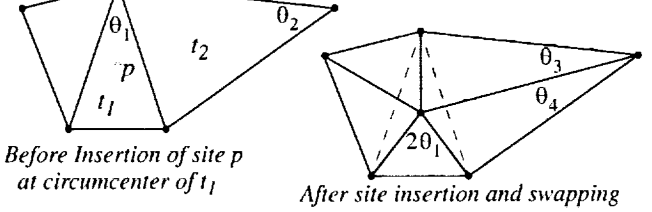


Figure 8. Quality manifold triangulation for manufacturing die example generated with Alg.M. in §3.3. Mesh improvement terminated after generating 2606 triangles when the minimum angle in the triangulation reached 25°. Chord-height and length-scale refinement not used.

gles (see the inset frames in Fig.8), and that the mesh length-scale varies smoothly over the part. The smallest angle in the mesh is 25.02°, which corresponds to a maximum aspect ratio of between 2.36 and 4.7 (by Eq.(3)).

While the histogram in Figure 7 shows a steady increase in the minimum angle, there is an irregular array of downward spikes in the profile. In the presentation of the PSLG algorithm, Ref.23 included a similar histogram, and noted the same characteristic. Consider the two triangles t_1 and t_2 shown at the left in Figure 9, Assume that θ_1 is the smallest angle in the mesh lying in triangle t_l. Furthermore, assume that t_1's face neighbor, t_2, has a small angle θ_2 opposite the shared edge which is only slightly larger than θ_1. Site p will get added at t_l's circumcenter improving θ_1 to $2\theta_1$. After application of the maxmin predicate $\tilde{\Phi}_{XN}$ on the shared edge, the swapped configuration at the right of Fig.9 may occur. This configuration includes two new triangles with a minimum angles θ_3 and θ_4 either of which may now be the smallest angle in the mesh and may actually be smaller than the original angle θ_1.

With this behavior understood, Figures 10 and 11 present angle histograms and example meshes for a more complicated assembly of parts. Alg.M was run on CAD parts for the main element of a transport wing, and a flap element for the



θ_1
θ_2
t_2
p
t_l
θ_3
θ_4
$2\theta_1$
Before Insertion of site p at circumcenter of t_l
After site insertion and swapping

Figure 9. Mechanism responsible for downward spikes in histogram of minimum angle as Alg.M proceeds. If θ_1 is initially the smallest angle in the mesh, angles θ_3 and θ_4 may be smaller after insertion of p and enforcement of Φ_{XN} by edge-swapping.

same wing. The main element consisted of 224 rational B-spline curves and 36 trimmed NURBS surfaces. The flap contained 31 rational B-spline curves and 10 trimmed NURBS surfaces.

The crosshairs on the curves in Fig.10 show that with a 25° angle bound, Alg. M will produce 20846 triangles on the main element and 15334 triangles on the flap. Figure 11 displays these triangulations.

The histograms in Figure 10 bear close resemblance to the one presented for the simple die example shown in Fig.7. All of these profiles are characterized by a sharp initial angle improvement and then a rolling-off as the minimum angle

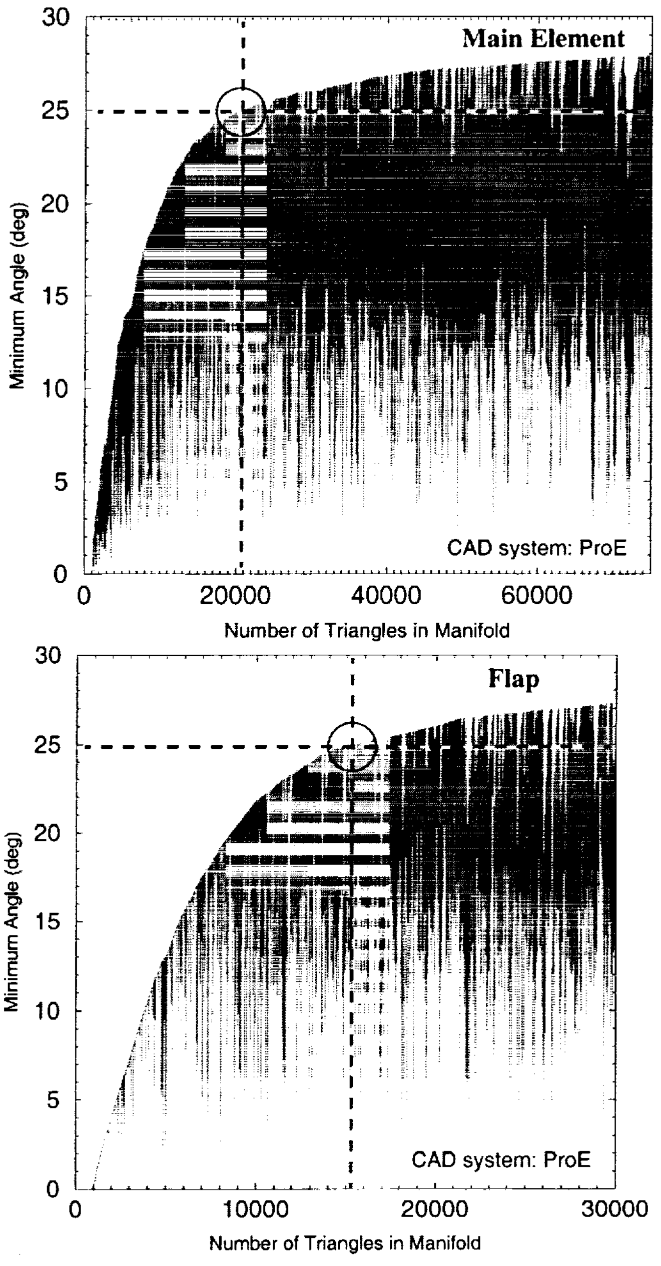


Figure 10. Angle histograms for Alg.M on the main element of a transport wing(upper), and on the main flap (lower). An angle bound of 25° would produce triangulations with 20846 and 15334 triangles on the main element and flap respectively.

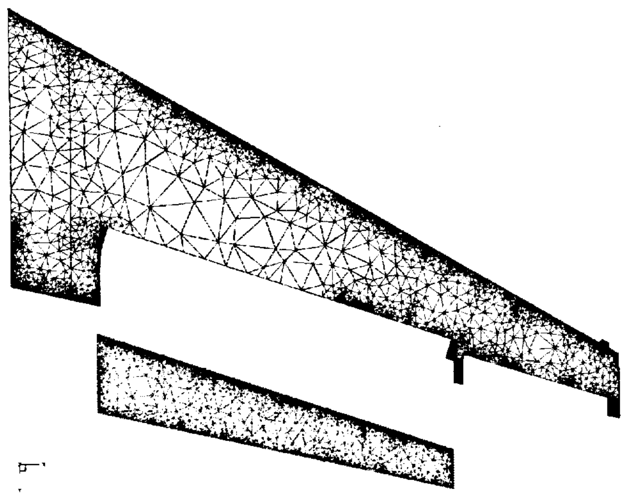


Figure 11. Bounded angle triangulations of main element of a transport aircraft wing and flap generated by Alg.M in §3.3. Minimum angle 25°, 20846 triangles on wing, 15334 triangles on flap.

climbs above about 20°. All three profiles exceed 27°, but reaching 30° seems unlikely.

The abscissa values in Fig.10 indicate relatively large triangulations as compared with the die example. The size-optimality property cited in §3.1 and §3.3 implies that triangle size must vary smoothly between different sized constraints. In both of these examples the smallest constraint in the CAPRI triangulation was a factor of 10^4 smaller than the largest constraining edge. In addition, practical experience with the algorithm indicates that larger initial triangulations, S_C, (Eq.(2)) have a proportionally slower initial rise in their angle profiles. This seems reasonable since if the CAPRI triangulation is complex there may initially be many bad angles which need improvement. Often CAD parts are unnecessarily complex for reasons dating to the specific events in their creation. Our experience with CAD repair software indicates that CAPRI generally produces less complex (sometimes by an order of magnitude) initial triangulations if the parts have been processed by CAD repair tools. Since tolerance to poor CAD parts is one type of robustness that we seek in this research, neither part shown in Fig.11 underwent such repair prior to creation of these triangulations.

With the minimum mesh angle restricted to 25°, triangles are again restricted to aspect ratio's less than 4.7. Such an isotropic mesh is very inefficient at meshing features with curvature in only one dimension. Therefore, if one intends to produce meshes for viscous computation, a substantially smaller angle bound may be appropriate.

4.2. Chord-Height Refinement

Enforcement of the angle bound does not guarantee that the edges are refined when they are sufficiently far from the underlying surface. Alg.E in §3.5 presented an edge refinement strategy for automatically breaking edges whose midpoints are far from the geometry. Figure 12 shows an interior corner of the die example after the imposition of a chord-height tolerance of 10^{-4} times L, where L is the maximum outer dimension of the part. Away from the curved interior corners, the triangulation remains unchanged since the faces of the die are planar.

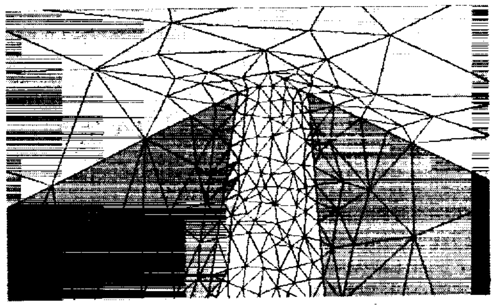


Figure 12. Interior corner of manufacturing die example triangulation after imposition of a chord-height tolerance of $1.x10^{-4}$ normalized by the maximum outer dimension of the part. Edge refinement was performed using Alg.E. in §3.5

5. Conclusions and Future Work

5.1. Conclusions

This paper examined the direct use of CAD geometry in surface meshing. It argued in favor of accessing CAD parts and assemblies in their native format, without translation and for the use of CAD-native predicates and constructors in mesh generation. The discussion also contended that since physical-space mesh generation techniques permit exact numerical computation, they can be more robust than their mapped-space counterparts.

The feasibility of this approach was demonstrated by application to a novel, physical-space, curved surface meshing algorithm to several CAD parts. Model data was accessed in its native format using the CAPRI API. This triangulation algorithm maintained a locally maxmin triangulation and used an approximate circumcenter site insertion strategy. All triangulation predicates were formulated for evaluation in physical-space, and examples demonstrated that the method produced a bounded aspect ratio manifold triangulation. The computational requirements for exact evaluation of all triangulation predicates were discussed. The algorithm was demonstrated on CAD parts of varying complexity and reliably produced triangulations with minimum angles in excess of 27°.

5.2. Future Research

• Although Alg.M. produces meshes with generally smooth length-scale variation, there is occasionally a discernible irregularity in the triangulations. This behavior becomes more pronounced when higher angle bounds are specified. Similar behavior has been noted in the PSLG algorithm [23], although in 2-D the behavior seems less pronounced. One possible source for this is the abrupt change in insertion location from circumcenter to centroid (of t_{opp}) if an obtuse triangle is encountered. Alternative strategies should be investigated since this behavior can degrade the efficiency of the triangulations.

• The initial triangulation returned by CAPRI depends strongly upon the "history" of the CAD part. Such effects are evident in the triangulation of the main wing element in Fig.11, for example, where the outboard portion of the flap cut-out is excessively resolved due to relics of the part's creation process. The only effective strategy we've found for avoiding this is to "clean" the geometry using CAD repair software. Alternative approaches should be investigated.

6. Acknowledgments

The CAPRI API was developed by R. Haimes under NASA Lewis Contract: NAG3-2019, his participation in this work is gratefully acknowledged. The author also extends thanks to M. Delanaye, T. Pulliam, Y. Liu and P. Walsh at NASA Ames for many productive and insightful discussions over the course of this work. Special thanks also go to J. Ruppert of Axys Pharmaceuticals for his comments on the quality surface meshing algorithm and insight into the PSLG method.

7. References

[1] Cosner, R. "Issues in aerospace applications of CFD analysis," *AIAA Paper 94-0464*, Jan 1994.

[2] Aftosmis, M.J., Berger, M.J., Melton, J.E., "Robust and efficient Cartesian mesh generation for component-based geometry." *AIAA Paper 97-0196*, Jan 1997.

[3] Löhner.R., "Finite element methods in CFD: Grid generation, adaptivity and parallelization." *AGARD Special course on unstructured Grid methods for Advection dominated Flows*, AGARD-R-787, May, 1992.

[4] Marcum, D.L., and Weatherhill, N.P., "Unstructured grid generation using iterative point insertion and local reconstruction." *AIAA Jol.* **33**(9), Sep. 1995.

[5] Pirzadeh, S., "Unstructured viscous grid generation by the advancing layers method." *AIAA Paper 93-3453-CP.* Jun. 1993.

[6] Wang, Z.J., Przekwas, A., and Hufford, G., "Adaptive Cartesian/adaptive prism grid generation for complex geometry," *AIAA Paper 97-0860*, Jan. 1997.

[7] ICEM CFD/CAE V.3.1.3: User Manual, Volume 2, Unstructured Grid Generation. ICEM Systems Inc. Berkeley CA, Nov. 1994.

[8] Bohn J.W., and Zozny, M.J., "Automatic CAD-model repair: Shell-closure." *Proc. Symp. On Freeform Fabrication*. Dept. of Mech. Eng., Univ. of Texas at Austin, 1992.

[9] Guéziec, A., Taubin, G., Lazarus, F., and Horn, W., "Cutting and stitching: Efficient conversion of a non-manifold polygonal surface to a manifold.", IBM-RC-20935. IBM Research Division, Yorktown Heights, NY, Jul. 1997.

[10] "VGRID: An unstructured tetrahedral grid generator based on the advancing front method." Vigyan Inc. Hampton VA. http://www.vigyan.com

[11] "CFD-GEOM Users Manual," CFD Research Corp., Huntsville Al, 1997.

[12] Reed, K., "The Initial Graphics Exchange Specification (IGES) Version 5.1", Sept. 1991.

[13] "Industrial automation systems and integration -- Product data representation and exchange -- Part 1:Overview and fundamental principles." *ISO/TR 10303-1.* International Standards Org. Genève, Switzerland, 1994.

[14] 3D Systems Inc. *Stereolithography Interface Format Specification*. 1988.

[15] Haimes, R., and Follen, G., "Computational Analysis PRogramming Interface." *Proceedings of the 6th International Conference on Numerical Grid Generation in Computational Field Simulations*, Eds. Cross, Eiseman, Hauser, Soni and Thompson, July 1998.

[16] Yap, C., "Robust geometric computation." *CRC Handbook of Discrete and Computational Geometry Eds.* Goodman J.E., and O'Rourke, J., CRC Press, Boca Raton Fl., 1997.

[17] Shewchuk, J.R., "Adaptive Precision Floating-Point Arithmetic and Fast Robust Geometric Predicates."*CMU-CS-96-140.* Carnegie Mellon Univ. School of Computer Science, May, 1996.

[18] Farin, G., *Curves and Surfaces for Computer Aided Geometry Design, A Practical Guide*, Academic Press, 1990.

[19] Michelucci, D. "The Robustness Issue." Internal report, Laboratoire d'Image de Synthèse de St Etienne, France. See http://www.emse.fr/~micheluc/english/michelucci.html

[20] Goldberg, D., "What every computer scientist should know about floating-point arithmetic." *ACM Comput. Surveys*, **23**(1):5-48, Mar. 1991.

[21] Bern, M., Eppstein, D. and Gilbert, J.R., "Provably good mesh generation." in *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*, **1**:231-241, IEEE,St. Louis, Missouri, Oct. 1990.

[22] Chew, L.P., "Guaranteed-quality triangular meshes." Technical Report, *TR-89-983*. Computer Science Dept, Cornell Univ. Apr. 1989.

[23] Ruppert, J., "A Delaunay refinement algorithm for quality 2-dimensional mesh generation." *Jol. of Algorithms*, **18**(3):548-585, May 1995.

[24] Chew, L.P., "Guaranteed-quality mesh generation for curved surfaces." *Proc. of the Ninth Annual Symposium on Computational Geometry*, 274-280. ACM, 1993.

[25] Green, P.J., and Sibson, R., "Computing the Dirichlet tessellation in the plane." *The Computer Journal*, **2**(21):168-173, 1977.

[26] Real/Expr homepage, 1996. Source code, documentation, examples and literature. http://simulation.nyu.edu/projects/exact.

[27] Ouchi, K., "Real/Expr: Implementation of an exact computation package", Master's Thesis, New York Univ., Dept. of Comp. Sci., Courant Inst. NY. Jan 1997.

[28] Trimaran homepage, 1998. http://react-ilp.nyu.edu.